LESSONS LEARNED ASSESSING SMALL COMPUTING PROGRAMS

Stephen Dannelly and Marguerite Doman
Department of Computer Science
Winthrop University
Rock Hill, SC

Abstract

Regardless of their size, computing programs are required to perform outcomes-based assessment by either their institution's regional accrediting body or for specialized program accreditation, such as ABET/CAC. Small programs face a unique set of problems including performing statistical analysis based on small numbers of students and placing all the burden of assessment activities onto a small faculty body. This paper describes some of the program assessment issues unique to small programs and suggests simple solutions to common problems.

Introduction

The need to perform program assessment is a reality for all computing programs at every US university and college. In the past, computer science faculty and administrators focused on topic coverage in the curriculum and the success of their graduates. Now all computer science programs must perform outcome-based program assessment for program accreditation and/or university accreditation. In the last decade, outcome-based program assessment became an important focus of ABET accreditation reviews. In the last few years, regional accrediting agencies have made assessment of all degree programs a priority when reaccrediting universities and colleges. There is no question as to whether or not a computer science program should perform program assessment. The question is how best to perform assessment.

Degree program assessment in small computing programs poses a unique set of problems. Small numbers of students makes it difficult to perform statistically significant analysis of student learning outcomes data. The complete array of assessment activities performed by a small faculty body, with no dedicated assessment staff, adds to a workload that already includes significant teaching loads and presents particular challenges in the management of program assessment. Yet, accrediting organizations give no special dispensation to small programs, nor should they.

Now in its 125th year, Winthrop University is a public comprehensive institution in South Carolina. Undergraduate fulltime enrollment is 4500, plus 650 part time students. Graduate enrollment is 1150. Most undergraduates live on campus or near the campus, which is dominated by grand oak trees and Neo-Georgian architecture.

The university offers three varieties of computing undergraduate degree programs. The BS in Computer Science (BSCS) has been accredited by ABET/CAC since 1990 and has about 75 students in the major. All computing courses in the Computer Information Systems (CIFS) option of the BS in Business are taught by the eight computer science faculty. The CIFS program is accredited by AACSB (business program accreditation) and has about 40 students. The BS in Digital Information Design (DIFD) combines computing coursework with graphic design and marketing coursework to focus on web application design. The DIFD program is not individually accredited because no accrediting body exists yet for such cross-discipline programs. DIFD is only four years old, has 80 majors, and is growing rapidly. Some computer science courses are shared among the three programs. For example, CIFS and BSCS majors take the same CS1 and CS2 courses. Some web application programming courses in the DIFD

program can be taken as electives by BSCS majors.

Each of those three degree programs has its own assessment program. Assessment of the three programs is the responsibility of the computer science faculty. In the case of DIFD, program assessment is coordinated with the Design (art) and Marketing departments. Managing assessment of three programs, one for ABET/CAC and one for AACSB (which use different vocabularies), and complying with Southern Association of Colleges and Schools (SACS) requirements for all three, has resulted in a computer science faculty that is constantly immersed in program assessment.

Because of the necessity and desire to focus on teaching and research and not be consumed by seemingly constant assessment reporting and site visits, we have developed processes that gather, analyze and act on rich assessment information, yet do not consume vast amounts of faculty time. Development of those processes has not always been smooth or easy.

The purpose of this paper is to explain some of the lessons learned about assessing small computing programs. Assessment issues that are common to both large and small programs are discussed first, followed by issues unique to small computer science programs. Finally, our program assessment methodology is outlined. Our processes are not perfect and not all aspects are transferable to other small institutions. But our processes have successfully met the challenges of several accreditation reviews.

Assessment Issues Common to both Small and Large Programs

The basics of assessment apply to small programs as much as large programs. Every program must have a public list of student learning outcomes, and in the case of ABET also a list of program objectives. Every program should use a variety of types of metrics. Every accrediting body stresses the importance of rubrics. But, while large programs have the resources to develop all these components from scratch and tailor these to their specific program, small degree programs need to borrow much of these assessment components from other institutions. Reinventing the wheel takes more time than small programs have to invest. In short, when beginning or improving an assessment process for a small computing program, research what others have done. When adopting tools and other materials, inform the source institution. The institution can then promote the dissemination of their work. Also, these metrics are validated to your accreditor as previously accepted assessment tools.

The number one rule for assessment, for large or small programs, is KISS. Keeping it simple can be difficult. Engineers and computer scientists love to devise complex and elaborate systems for measuring experiments. But, to paraphrase assessment guru Gloria Rogers, it is okay to assess a degree program with “two sticks and a chain instead of calipers”. Two sticks separated by about 10 yards of chain that are placed where a guy in a striped shirt has put his foot is good enough to determine if highly paid professional football players have progressed a ball far enough down a field. Focus on what assessment is trying to determine – are we generally getting closer to our goal? An elaborate methodology composed of multilayered analysis of detailed measurements is very likely to be overkill.

Small programs and large programs can lean heavily on measurements already existing in their programs. Every professor has course goals and gives exam questions, lab assignments, or projects to determine if students are meeting those learning goals. The general relationships between course goals and program outcomes are usually illustrated as a matrix. Where course goals overlap with program outcomes there are possible program assessment measurements [1]. If program outcomes include communication skills, then the course(s) that

cover communication skills are likely to be already assessing student's communication skills in some way.

We do not have to measure everything about the program. Determine the essential program outcomes then find a few (two to four) ways to easily and fairly accurately assess those outcomes. Use a variety of measurement types. Results of projects and exam questions are easy to gather. Do not use course grades, because the focus of any course is unlikely to directly align with the more general program outcomes. Do not rely heavily on indirect measurements, because we need to know what students actually know and can do, not what they think they know. However, alumni and employer surveys are valuable. Not every student needs to score perfectly for overall student performance to meet expectations. External assessments, such as ETS' Major Field Test, provide a good degree of external validation.

Problems Unique to Small Programs

One of the biggest obstacles to assessment for large and small programs is sometimes faculty. Specifically, the "established" faculty members that believe assessment is yet another fad. They have not jumped on every bandwagon over the decades and don't intend to start jumping now. In a large program, such a senior member is likely not teaching many courses and may be easily side-stepped. But in a small program, everyone's help is needed and so everyone must buy into the need to perform assessment.

Another set of problem faculty members is the group that misunderstands the purpose of assessment. A misguided faculty member who is told to report specific student performance data to their department chair could believe the data will be used to assess the faculty member's performance in addition to the program's learning outcomes. When beginning a formal assessment program it is essential to clearly communicate to professors the importance of the program assessment, how data will be used, and how data will not be used. Some educators, such as Rigby and Dark [2], use assessment data taken from an individual course to improve instruction in that course. We believe program assessment should attempt to identify the strengths and weaknesses of the overall program, not assess the teaching in individual courses. When closing the assessment loop, a program improvement action plan is created for each program weakness that has been identified. Of course, those plans always include changes to one or more specific courses. But the course changes are made in the context of the program's objectives and outcomes and the degree's curriculum, not one instructor.

A significant problem for small programs is assessing student performance when there are small numbers of students. Good statistical analysis relies on a sufficient sample size (N value). A large program on a six-year ABET cycle might only need to gather assessment data every three years. That is, we will access one third of our student learning outcomes this year, then next year a different third of our outcomes, and so on. So, in the six years, each metric receives two significant reviews.

But, small programs don't have the luxury of waiting. Small programs need to assess every semester. Some core courses may be offered only once a year with an enrollment of 10 or fewer students. If the expected performance level is "80% of students will score at least 70 out of 100 on this project", then when two out of nine students earn a D+, then the whole group fails to meet performance expectations. So, one or two poor students at the wrong time can incorrectly reflect the program's attainment of outcomes. The solution to small N values is to gather assessment data every semester, then aggregate the scores over two or three years. Of course, if 4 out of 18 students in two years, or 6 of 27 students in three years, are not meeting expectations, the poor performance is not an anomaly.

We learned this lesson after just two years of using the Major Field Test. Only five students took the exam the year we started using the

exam. For reasons described in the next section, the program's results were poor - only the 20th percentile in the three major exam categories. The next year twelve students took the exam and the program scored in the 80th percentile. What had we done to improve the program? The answer is nothing significant. That second group of test takers happened to include several very bright seniors. Eight of the twelve are now in PhD programs. The next year we dropped to the middle of the percentile brackets, but the program had not declined in quality. We now combine MFT results across multiple years. Year to year variations in MFT results usually correlate with variations in our internal data, thus helping to validate our methods. Using the MFT for such validation is not unusual [3].

Small programs that offer each core course once a year can also suffer from putting all their eggs in one professor's basket. A course-embedded metric that comes from one professor's course is dependent on that professor. If that professor is a tough grader, or simply unsuccessfully tries something new one semester, then a year's worth of data for that metric will suffer. Rubrics are used by large programs to standardize results across course sections and professors. Rubrics can be used by a small program to guide an individual professor and standardize results from year to year.

In a large institution it makes economic sense to employ fulltime staff to guide and manage assessment activities. Simply gathering, organizing, cleaning, and archiving data can be very time consuming. Small programs do not have any such staff. It is up to the faculty members to not only perform the high level analysis work, but also the low level work. Since the bulk of assessment data gathering is likely to be tied to specific courses, and hence becomes the responsibilities of the course instructor, we recommend all course syllabi be required to contain a short assessment statement that specifies what data, if any, is collected for assessment purposes. The main purpose of that requirement is to remind the faculty member to perform the assessment.

Lack of dedicated assessment staff also increases the chances something will slip through the cracks. Faculty members rightly focus on their courses and their research, not assessment. To insure activities not associated with courses are conducted each year or semester, we highly recommend an annual calendar of events and deadlines be created. And to make sure the tasks get done, put a name next to each task. Our college's faculty annual report includes a section where each faculty member must report on their program assessment activities, thus assessment is incorporated into the annual faculty review process.

Our Methodology

Our assessment procedures have evolved slowly over many years. When ABET began making assessment an important provision many years ago, our department developed a very thorough and very formal process for assessing many aspects of the BS in Computer Science degree program. To be frank, it was too big a beast to be sustainable. After that review cycle, we learned to rely more on activities we were naturally performing to assess our students. We soon found that the new collection of fewer and more straightforward metrics was pointing in the same direction as the previous large and cumbersome set of metrics. The objective of our current assessment process is to provide good visibility into student performance, while making assessment activities routine and minimally invasive to professors' academic freedom.

Much of the evolution of our assessment processes has been driven by ABET accreditation standards. But new SACS requirements have caused us to recently modify our approach. For example, SACS now requires all universities to have College Level Competencies (SACS requirement 3.5.1 to be

specific) - student learning outcomes for the general education program. At our university, that has meant that every degree program must now also assess the university's four competencies. The university's General Education program assesses all freshmen and sophomores, then we assess our juniors and seniors at a deeper level. With this additional assessment requirement, the computer science program decided to tweak our lists of student learning outcomes to overlap the four university competencies. That way, as we assess our outcomes, we can simply report a subset of the results to the university administration as our assessment of their four competencies.

As with all degree programs everywhere, our program objectives and student learning outcomes are derived from our program's mission and the university's mission, with input from our constituents. Our advisory board is composed of successful alumni and of employers who regularly recruit our students. The BS in Computer Science's list of nine student learning outcomes includes the usual themes of communication, leadership, problem analysis, and theoretical foundations of computer science. The CIFS program has just three student learning outcomes because CIFS is part of the BS in Business degree program that has its own five outcomes. The CIFS list overlaps the BSCS list, thus simplifying the gathering of course-embedded data. The DIFD degree program has four student learning outcomes.

Each student learning outcome is measured three ways. Our chosen collection of metrics is primarily composed of course-embedded metrics. Data include grades on specific types of projects such as grades on the design of a compiler, grade averages of programming assignments in Data Structures, and grades on different types of presentations and reports. Other course-embedded metrics include a non-graded assessment of teamwork skills performed by the instructor of the senior projects course. As much as possible, we selected metrics that leaned toward the upper end of Bloom's taxonomy; i.e., more synthesis and evaluation, and less knowledge and comprehension.

Choices of metrics abound [4]. Some programs integrate course surveys [5], while others use student focus groups [6]. A recent trend in program assessment is the use of portfolios. Clemson University even requires all undergraduates to submit items to an electronic portfolio that is used to assess the university's core competencies [7]. However, we do not use portfolios. While portfolios would be an improvement to our assessment methodology, that addition would simply not have a positive return on the investment of designing the process, integrating the technology, and regularly assessing the portfolios.

Our non-course-embedded metrics are few in number, but not lesser in significance. We use the ETS Major Field Test for Computer Science as part of the BS in computer science degree program assessment. For a small program, a small supply of exam booklets lasts a few years. So, that exam is a manageable item on the budget. When we began using the MFT our problem was how to get students to take the test. Carrots, such as a few bonus points in the senior projects course, were not sufficient to entice enough students to take a two hour exam. Additionally, the top performing students did not need the extra credit, so results were skewed downward. A stick ended up working better than carrots. We added a graduation requirement that every student must complete an assessment exam in their final semester. Encouraging students to do their best and take the exam seriously has never been a problem for us. Before the exam begins, the proctor explains that the exam is used in the accreditation process for their degree and the students' performance will be compared to students at other universities. Students are competitive and want their institution to do well.

Our assessment for all three computing degrees is performed on an annual cycle.

Course data is collected each semester. Non course data, such as the MFT exam, is gathered each spring semester. Data is summarized in early summer. By October of each year the faculty meet to discuss the prior year's results. Action plans to address weaknesses are then begun, to be reviewed the next October. Subsets of this information are submitted to the university, for SACS purposes, each February.

The first task on the annual schedule is for the department chair to remind all instructors of what assessment data needs to come from what courses. In mid-August, when syllabi are being written, the department chair emails a single page summary of assessment needs. The list of metrics is organized by course. So, the Data Structure professor knows at the end of the semester he will report two items: average of all programming assignments (and there must be at least seven programs assigned), and the grade of a project that involves designing a solution to a complex open-ended problem.

Course embedded data are emailed to the department chair at the end of each semester as Excel files. Faculty only report data for the majors in question; CS majors for the CS assessment, CIFS majors for the CIFS assessment, etc. So the math majors in CS2 are deleted before the faculty sends in the data. The files are archived by academic year and named according to the course. The department chair processes the data in early summer. He determines if performance met expectations. For example, did 70% of CS majors in Data Structure score 70 or higher on the project? These results are entered into a grid. We do this processing manually. While several universities have developed very good semi-automated data processing systems and such systems are necessary when processing 400 students, using such a system to process 15 grades in a data structures course or 10 peer evaluations in a senior capstone course does not merit that investment for us.

Near the beginning of October, the department meets to discuss the prior year's results. We do not have a departmental curriculum or departmental assessment committee. In a department of 10 faculty members, the entire 10 would simply rehash whatever results any subgroup derived. So, all 10 look at the results and identify any areas of weakness.

Program weaknesses are addressed through the development of Curriculum Improvement Plans (CIPs). A small group of faculty interested in the weakness decide on the steps necessary to address the problem. For example, if communication results have dipped to "does not meet expectations" for two straight years, then the professors that teach courses that emphasize communication skills determine how to improve communication instruction. Most importantly, they write down their plan in a standardized short format. Such documentation is critical for degree programs to prove to site visit teams that the program "closes the loop" on assessment. It is not sufficient to say, "here are our goals, here is how we determine if we are meeting goals, and here is how we are doing." A program must also say, "And here is what we have done to improve." CIPs are reviewed the next October to determine their effectiveness. Supporting documentation, such as pre and post syllabi, are archived with the CIP.

Conclusions

Outcomes-based program assessment is a valuable activity. But in a small computing degree program it is yet another activity that consumes significant amounts of already scarce time. Therefore, sticking with the KISS principle is essential. The best use of time when assembling the assessment elements, such as grading rubrics, review guidelines and surveys, is modifying the elements of other institutions' assessment programs - it is faster to modify someone else's wheel than to reinvent it yourself. Small programs must leverage their already existing assessments of students' learning when assessing the program's student

learning outcomes. Faculty buy-in is necessary to distribute the workload. Simple steps such as including assessment needs on syllabi and a calendar of annual deadlines and review meetings help institutionalize the assessment process. A well thought-out annual assessment time-line that relies heavily on existing measurements of student learning is likely more than sufficient for accrediting agencies. In short, simple does not equate to insufficient.

References

1. Jenq-Foung Yao, Yi Liu, Autumn Grubb, and Gita Williams, "Course assessment framework that maps professional standard and ABET accreditation criteria into course requirements", *Journal of Computing in Small Colleges*, December 2007.

2. Steve Rigby and Melissa Dark, "Using Outcomes-Based Assessment Data to Improve Assessment and Instruction: A Case Study", *ACM SIGITE Newsletter*, Vol. 3, No. 1, January 2006.

3. Robert Jarman and Sankara N. Sethuraman, "A Pilot Study for Developing a Quantitative Model for Outcomes Assessment in the Computer Science Program at a Small University", *Journal of Computing in Small Colleges*, January 2001.

4. Kathryn E. Sanders and Robert McCartney "Program Assessment Tools in Computer Science: A report from the Trenches", *Proceedings of the 34th SIGCSE technical symposium on Computer science education* (SIGCSE '03), 2003 .

5. Kwok-Bun Yue, "Effective Course-Based Learning Outcome Assessment for ABET Accreditation of Computing Programs", *Journal of Computing in Small Colleges*, April 2007.

6. Dick Blandford and Deborah Hwang, "Five Easy But Effective Assessment Methods", *ACM SIGCSE Bulletin*, Vol 35 Issue 1, January 2003.

7. Clemson University, http://www.clemson.edu/academics/programs/eportfolio/, (accessed April 2011).

Biographical Information

Dr. Stephen Dannelly has been chair of the Department of Computer Science at Winthrop University since joining the university in 2005. His research and teaching specialize in the area of software project management.

Marguerite Doman is an Assistant Professor at Winthrop University. She received her PhD from the University of North Carolina at Charlotte in 2009. Before joining academia, she worked as a systems programmer at IBM in operating systems development. Her research interests include computer science education and networking.